

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-49370

(43) 公開日 平成10年(1998) 2月20日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/38	3 1 0		G 0 6 F 9/38	3 1 0 F

審査請求 未請求 請求項の数6 O L (全 20 頁)

(21) 出願番号 特願平8-203675

(22) 出願日 平成8年(1996) 8月1日

(71) 出願人 000006013

三菱電機株式会社

東京都千代田区丸の内二丁目2番3号

(72) 発明者 エドガー・ホルマン

東京都千代田区丸の内二丁目2番3号 三
菱電機株式会社内

(72) 発明者 吉田 豊彦

東京都千代田区丸の内二丁目2番3号 三
菱電機株式会社内

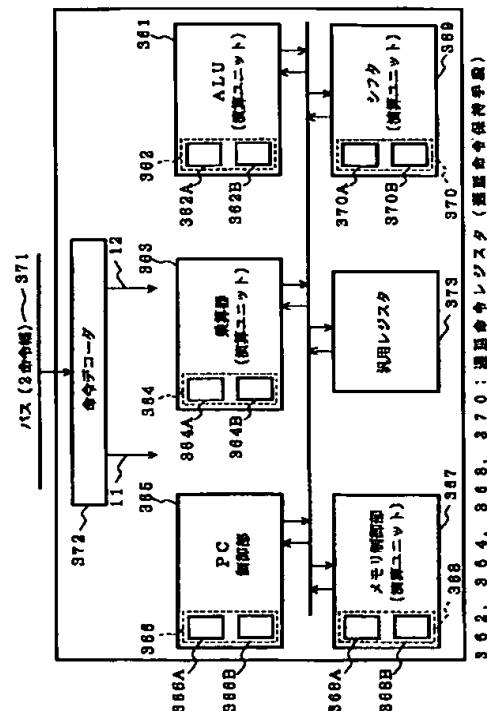
(74) 代理人 弁理士 田澤 博昭 (外2名)

(54) 【発明の名称】 遅延命令を有するマイクロプロセッサ

(57) 【要約】

【課題】 遅延命令は分岐命令に限られているので、効果的な命令のスケジューリングができないという課題があった。

【解決手段】 ALU361、乗算器363、PC制御部365、メモリ制御部367およびシフタ369は、遅延命令のデコード結果をレジスタ362A、364A、366A、368A、370Aに格納するとともに、遅延命令で指定された遅延量に応じたプログラムカウンタ値をレジスタ362B、364B、366B、368B、370Bに格納するものである。



【特許請求の範囲】

【請求項1】 命令をデコードする命令デコーダと、前記命令デコーダの出力に従って命令を実行する命令実行部とを備えた遅延命令を有するマイクロプロセッサにおいて、前記命令実行部におけるプログラムカウンタ値を制御するPC制御部は、遅延分岐命令の分岐先を示す値および遅延分岐命令で指定された遅延量に応じたプログラムカウンタ値を保持する遅延分岐命令保持手段を備えたことを特徴とする遅延命令を有するマイクロプロセッサ。

【請求項2】 命令をデコードする命令デコーダと、前記命令デコーダの出力に従って命令を実行する命令実行部とを備えた遅延命令を有するマイクロプロセッサにおいて、前記命令実行部における演算ユニットは、遅延演算命令による演算内容および遅延演算命令による演算の実行開始を示す値を保持する遅延命令保持手段を備えたことを特徴とする遅延命令を有するマイクロプロセッサ。

【請求項3】 遅延命令保持手段は、遅延演算命令で指定された遅延量に応じた値を保持することを特徴とする請求項2記載の遅延命令を有するマイクロプロセッサ。

【請求項4】 遅延命令保持手段は、遅延演算命令で指定された遅延量に応じたプログラムカウンタ値を保持することを特徴とする請求項3記載の遅延命令を有するマイクロプロセッサ。

【請求項5】 各演算ユニットは、複数の遅延命令保持手段を有する請求項3または請求項4記載の遅延命令を有するマイクロプロセッサ。

【請求項6】 命令実行部は、複数の演算を同時に実行する請求項1から請求項5のうちのいずれか1項記載の遅延命令を有するマイクロプロセッサ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、実行時期を遅らせて実行される遅延命令を有するマイクロプロセッサに関するものである。

【0002】

【従来の技術】図20は従来のパイプライン制御方式によるマイクロプロセッサの処理シーケンスを示すシーケンス図である。図において、300は分岐命令、301はパイプラインにおける命令フェッチステージ、302はパイプラインにおけるデコードステージ、303はパイプラインにおける命令実行ステージ、304はパイプラインにおけるライトバックステージ、305は第1の遅延スロットにおける命令、306は第2の遅延スロットにおける命令、307は分岐先で実行される命令を示す。なお、以下、単に「分岐命令」と表現された場合には、それは、プログラムカウンタ(PC)値にオペランドで示されるオフセット値が加算された値であるアドレスに分岐する命令と、オペランドで示されるアドレスに

間接的にまたは直接に分岐する命令とを含む。分岐命令とジャンプ命令とが、併記された場合には、それらは、それぞれ、プログラムカウンタ値にオペランドで示されるオフセット値が加算された値であるアドレスに分岐する分岐命令、オペランドで示されるアドレスに間接的にまたは直接に分岐するジャンプ命令を意味する。また、以下、分岐命令およびジャンプ命令は、サブルーチンコール命令を含むものとして説明を進める。

【0003】次に動作について説明する。分岐命令が実行されると、図20に示されたパイプライン制御を行うマイクロプロセッサは、分岐先アドレスを3番目のステージである命令実行ステージ303で得ることができる。その時点では、第1の遅延スロットにおける命令305および第2の遅延スロットにおける命令306は、既にデコードステージ302または命令フェッチステージ301にある。マイクロプロセッサは、それらの命令を無効なものとして扱わなくてはならず、パイプラインの無駄が生ずる。

【0004】パイプラインの無駄を防止するために、「コンピュータアーキテクチャ：量的アプローチ(Computer Architecture: A Quantitative Approach), Morgan Kaufmann社, 1990年」等の文献で種々の手法が提案されている。例えば、命令のスケジューリング、または命令のスケジューリングと遅延分岐命令(ディレイド分岐命令)との組み合わせによって、パイプラインの無駄を低減することができる。

【0005】例えば、特開平6-131180号公報や特開平6-274352号公報に、ディレイド分岐命令に関する技術が記載されている。一般にディレイド分岐命令における遅延量はマイクロプロセッサのアーキテクチャに応じた固定的な値であるが、特開平6-131180号公報には任意の遅延スロット数を指定できる命令が開示されている。指定された遅延スロット数はデクリメントカウンタに設定される。デクリメントカウンタの値は動作クロック信号の入力に従って減り、値が「1」になると分岐先命令のフェッチが開始される。

【0006】図21は2演算を同時に行うマイクロプロセッサにおける命令デコーダおよび命令実行部の部分の一般的な構成を示すブロック図である。図において、341は算術論理演算を実行するALU、342は乗算を実行する乗算器、343はPC値を計算するPC制御部、344はアドレス計算を行うメモリ制御部、345はシフト演算を実行するシフト、346は1サイクルで2命令を転送できるバス、347は命令をデコードして命令実行部に制御信号11、12を与えるデコーダ、348は汎用レジスタである。

【0007】図22はプログラムの一例を示す説明図である。図において、ADD, SRA, SUB, MULおよびJMPは、それぞれ、加算命令、シフト命令、減算

命令、乗算命令およびジャンプ命令を示す。加算命令、シフト命令、減算命令、乗算命令およびジャンプ命令は、それぞれ、命令実行部におけるALU341、シフタ345、ALU341、乗算器342およびPC制御部343において扱われる。また、例えば、(r3, r0, 6)は、汎用レジスタ348中のr0レジスタの値と即値「6」とを対象とした演算結果を汎用レジスタ348中のr3レジスタに設定することを示す。

【0008】図20に示されたマイクロプロセッサは2演算命令を扱えるので、命令実行部におけるリソースが競合しない限り、図22に示されたプログラムを2演算命令による各命令に変換したものを扱える。例えば、図23に示されるように変換された各命令を実行できる。図23において、各行は2演算命令の1命令に対応する。すなわち、各行は同時実行される2演算を示す。SRAとSUBとは、リソースの競合はないがレジスタ依存の関係にあるので、同時に実行することはできない。よって、2行目にNOPが置かれている。このような命令のスケジューリングは、プログラマまたはコンパイラによってなされる。

【0009】

【発明が解決しようとする課題】従来の遅延命令を有するマイクロプロセッサは以上のように構成されているので、以下のような課題があった。

(1) 遅延量が固定的にしか指定できないディレイド分岐命令を有するマイクロプロセッサでは、効果的な命令のスケジューリングができない。例えば、図23における5行目のJMPを削除して2行目のNOPの位置に遅延量「2」のディレイドジャンプ命令を置くことは効果的である。そのようにすれば、図23の4行目の命令のフェッチ後に直ちにジャンプ先アドレスTGTの命令のフェッチが開始され、パイプラインの無駄が生じないからである。遅延量「2」しか指定できないマイクロプロセッサでは、そのようなディレイド分岐命令を置いた命令群を実行できる。しかし、例えば遅延量「3」しか指定できないマイクロプロセッサを使用する場合には、プログラマまたはコンパイラは、そのようなスケジューリングを行うことはできない。

(2) 指定された遅延量に応じた値をデクリメントカウンタに設定するマイクロプロセッサでは、命令遅延中に割り込みや新たな分岐が生じた場合に矛盾なく処理を進めるための構成が複雑になる。例えば、デクリメントカウンタの値は動作クロックに従ってカウントダウンされるので、何等の考慮も払わないと、設定された遅延量と実際の遅延量との間で、割り込み処理等で費やされた動作クロック数分の狂いが生ずる。

(3) ディレイド命令は分岐命令に限られているので、効果的な命令のスケジューリングができない。

【0010】この発明は上記のような課題を解決するためになされたもので、処理の矛盾を生じさせることなく

任意の遅延量を指定できるディレイド命令を扱えて効果的な命令のスケジューリングができる環境をプログラマに提供でき、その結果、プログラムをより高速に実行しうるマイクロプロセッサを得ることを目的とする。

【0011】

【課題を解決するための手段】請求項1記載の発明に係る遅延命令を有するマイクロプロセッサは、PC値を制御するPC制御部が、ディレイド分岐命令の分岐先アドレスを示す値およびディレイド分岐命令で指定された遅延量に応じたプログラムカウンタ値を保持する遅延分岐命令保持手段を有するものである。

【0012】請求項2記載の発明に係る遅延命令を有するマイクロプロセッサは、各演算ユニットが、ディレイド演算命令の内容およびディレイド演算命令の固定のまたは可変の遅延量に応じた値を保持する遅延命令保持手段を有するものである。ここで、ディレイド演算命令の内容とは、ディレイド演算命令のデコード結果であって、いかなる演算を行うかを示す情報である。なお、特に断らない限り、ディレイド分岐命令は、ディレイド演算命令に含まれないものとする。

【0013】請求項3記載の発明に係る遅延命令を有するマイクロプロセッサは、任意の遅延量を指定できるディレイド演算命令で指定された遅延量に応じた値を保持する遅延命令保持手段を有するものである。

【0014】請求項4記載の発明に係る遅延命令を有するマイクロプロセッサは、各演算ユニットが、ディレイド演算命令で指定された遅延量に応じたPC値を保持する遅延命令保持手段を有するものである。

【0015】請求項5記載の発明に係る遅延命令を有するマイクロプロセッサは、各演算ユニットに、複数の遅延命令保持手段が設けられているものである。

【0016】請求項6記載の発明に係る遅延命令を有するマイクロプロセッサは、上記の各構成を有するとともに、命令実行部が複数演算を同時に実行するものである。

【0017】

【発明の実施の形態】以下、この発明の実施の一形態を説明する。

実施の形態1. 図1はこの発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。このマイクロプロセッサは、32ビットの内部データバスを有する32ビットマイクロプロセッサである。図において、2は命令RAM6から64ビット幅のIDバスを介して入力した命令コードをデコードする処理を行う命令デコードユニット(命令デコーダ)、3はアドレス計算を行うメモリユニット(命令実行部)、4は論理演算やシフト演算を行う整数演算ユニット(命令実行部)、5は32ビット×64ワードの汎用レジスタ、7はデータが格納されるデータRAMである。

【0018】命令デコードユニット2において、8, 9

はそれぞれ命令コードをデコードするデコーダ、10はプロセッサの状態を示すプロセッサ状態語 (Processor Status Word、以下、プロセッサ状態語をPSWと呼ぶ)である。命令デコードユニット2は、さらに、デコーダ8のデコード結果とPSW10の内容にもとづいて制御信号11を作成し、それをメモリユニット3に与える。また、命令デコードユニット2は、デコーダ9のデコード結果とPSW10の内容にもとづいて制御信号12を作成し、それを整数演算ユニット4に与える。

【0019】メモリユニット3において、13はジャンプや分岐を含まない命令を実行するとPC値に8を加えて次に実行する命令に対するPC値を算出するとともに、ジャンプや分岐を含む命令の実行時に分岐変位をPC値に加算したり、演算で指定されたアドレッシングモードに応じた計算を行ってジャンプ先の命令に対するPC値を計算するPC制御部である。また、PC制御部13は、計算したPC値を32ビット幅のIAバスを介して命令RAM6に送り、命令RAM6から命令コードを出力させる。14はオペランドとなるデータのアクセスを制御するメモリ制御部である。メモリ制御部14は、32ビット幅のDAバスを介してアドレスデータをデータRAM7に転送し命令実行に必要なデータを64ビット幅のDDバスを介してアクセスする。15は汎用レジスタ5から32ビット幅のS1バス、S2バス、S3バスを介して転送された最大3ワードのデータを用いて算術論理演算を行い演算結果を32ビット幅のD1バスを介して汎用レジスタ5に転送するALU、16は汎用レジスタ5からS1バス、S2バス、S3バスを介して転送されたデータを用いてシフト演算を行い演算結果をD1バスを介して汎用レジスタ5に転送するシフトである。

【0020】メモリユニット3に対して、S1バス、S2バス、S3バス、S4バスを介して、32ビット長のデータを一時に4ワード転送することが可能である。従って、例えば、第1のレジスタの内容と第2のレジスタの内容との和でアドレッシングされるメモリの領域に第3のレジスタの内容をストアするとともに、第3のレジスタの内容をストアしたアドレスに所定値を加算して得られる値でアドレッシングされるメモリの領域に第4のレジスタの内容をストアする2ワードストア命令を実現することができる。また、メモリユニット3は、D1バスおよびD2バスを介して、メモリユニット3内の2ワードの演算結果またはデータRAM7から転送された*

* 2ワードのデータを汎用レジスタ5に転送することができる。

【0021】整数演算ユニット4において、17は汎用レジスタ5から32ビット幅のS4バス、S5バス、S6バスを介して転送された最大3ワードのデータを用いて乗算を行い演算結果を32ビット幅のD2バス、D3バスを介して汎用レジスタ5に転送する乗算器、18は乗算の結果を累積加算または累積減算して保持するアキュムレータである。アキュムレータとして、64ビットのものが2本ある。19は汎用レジスタ5からS4バス、S5バス、S6バスを介して転送された最大3ワードのデータを用いて算術論理演算を行い演算結果をD2バス、D3バスを介して汎用レジスタ5に転送するALU、20は汎用レジスタ5からS4バス、S5バス、S6バスを介して転送されたデータを用いてシフト演算を行い演算結果をD2バス、D3バスを介して汎用レジスタ5に転送するシフトである。

【0022】このマイクロプロセッサでは、汎用レジスタ5から、最大6種類のレジスタ値を読み出すことが可能であって、読み出されたデータは、それぞれ、S1バス、S2バス、S3バス、S4バス、S5バス、S6バスに出力される。また、汎用レジスタ5には、D1バス、D2バス、D3バスを介して最大3種類のレジスタ値を同時に書き込むことが可能である。

【0023】図2はこのマイクロプロセッサの命令フォーマットを示す説明図である。命令フォーマットとして、図2(a)に示すような1つの命令コードで2つの演算(operation)を指示する2演算命令のフォーマット101と、図2(b)に示すような1つの命令コードで1つの演算を指示する1演算命令のフォーマット102とがある。2演算命令のフォーマット101には、フィールド103およびフィールド104からなるフォーマットフィールドと、2つの演算フィールド106、107と、各演算フィールド106、107に付属する各実行条件フィールド105とがある。1演算命令のフォーマット102には、フィールド103およびフィールド104からなるフォーマットフィールドと、演算フィールドと、演算フィールドに付属する実行条件フィールド105とがある。演算フィールドは、フィールド108、109、110からなる。

【0024】フォーマットフィールドは、以下のような意味を示す。

コード：フォーマット

実行の順番

	operation_0	operation_1
FM=00: 2命令	第1	第1
01: 2命令	第1	第2
10: 2命令	第2	第1
11: 1命令	第1

ここで、FMは、フィールド103およびフィールド1※50※04からなる2ビットの値である。

【0025】FM=00の場合、この命令は2演算命令であることを示す。そして、演算フィールド106で指示されたoperation_0の演算と演算フィールド107で指示されたoperation_1の演算とが、デコード直後のクロックサイクルで並列に実行される。operation_0の演算はメモリユニット3で実行され、operation_1の演算は整数演算ユニット4で実行される。FM=01の場合、この命令は2演算命令であることを示す。そして、operation_0の演算が、デコード直後のクロックサイクルで実行され、operation_1の演算が、operation_0の演算に対して、1クロックサイクル遅れて実行される。FM=10の場合、この命令は2演算命令であることを示す。そして、operation_1の演算が、デコード直後のクロックサイクルで実行され、operation_0の演算が、operation_1の演算に対して、1クロックサイクル遅れて実行される。FM=11の場合、この命令は1演算命令であることを示す。そして、フィールド108、109、110からなる演算フィールドで指示された1つの演算がデコード直後のクロックサイクルで実行される。

【0026】実行条件フィールド105は、以下のよう
な意味を持つ。

コード: 実行条件
CC=000: 常時
001: F0=真 かつ F1=無視
010: F0=偽 かつ F1=無視
011: F0=無視 かつ F1=真
100: F0=無視 かつ F1=偽
101: F0=真 かつ F1=真
110: F0=真 かつ F1=偽
111: 予約済

【0027】実行条件フィールド105は、マイクロプロセッサの実行コントロールフラグF0、F1の値に応じて、演算フィールド106、107のoperation_0の演算やoperation_1の演算、およびフィールド108、109、110からなる演算フィールドの演算が有効であるか無効であるか定める。実行コントロールフラグF0、F1については後で説明する。演算が有効であるとは、演算結果がレジスタ、メモリおよびフラグに反映され、その演算による動作の結果が残ることを意味する。演算が無効であるとは、演算結果がレジスタ、メモリおよびフラグに反映されず、あたかも無効演算(NOP)が実行されたかのような動作の結果が残ることを意味する。

【0028】実行条件フィールド105の値CC=000のときには、実行コントロールフラグF0、F1の値にかかわらず常に演算は有効である。CC=001のときには、実行コントロールフラグF0=真のときにのみ演算は有効である。実行コントロールフラグF1の状態はいずれでもよい。CC=010のときには、実行コントロールフラグF0=偽のときにのみ演算は有効であ

る。実行コントロールフラグF1の状態はいずれでもよい。CC=011のときには、実行コントロールフラグF1=真のときにのみ演算は有効である。実行コントロールフラグF0の状態はいずれでもよい。CC=100のときには、実行コントロールフラグF1=偽のときにのみ演算は有効である。実行コントロールフラグF0の状態はいずれでもよい。CC=101のときには、実行コントロールフラグF0=真かつF1=真のときにのみ演算は有効である。CC=110のときには、実行コントロールフラグF0=真かつF1=偽のときにのみ演算は有効である。CC=111のときの動作は未定義であり、ユーザは、CC=111となる命令を用いることはできない。

【0029】図3は演算フィールドの詳細な内容を示す説明図である。フォーマット111~117は、それぞれ28ビットで表現される短型の演算フィールド106または演算フィールド107によるものである。フォーマット118は、フィールド108、109、110で構成される長型の演算フィールドによるものである。

【0030】フォーマット111(Short_M)は、演算内容を指定するフィールド120、レジスタ番号を指定する2つのフィールド121、122、レジスタ番号または6ビット長の即値を指定するフィールド123、およびフィールド123がレジスタ番号を示すのか即値を示すのかを指定するフィールド124で構成される。図3に示すように、フィールド124の値Xが「00」、「01」または「11」であるときにはフィールド123がレジスタ番号を示していることを示し、「10」であるときには即値を示していることを示す。

このフォーマット111は、レジスタ間接アドレッシングのメモリアクセス演算に用いられる。

【0031】フォーマット112(Short_A)は、演算内容を指定するフィールド120、レジスタ番号を指定する2つのフィールド121、122、レジスタ番号または6ビット長の即値を指定するフィールド123、およびフィールド123がレジスタ番号を示すのか即値を示すのかを指定するフィールド125で構成される。図3に示すように、フィールド125の値X'が「0」であるときにはフィールド123がレジスタ番号を示していることを示し、「1」であるときには即値を示していることを示す。このフォーマット112は、算術演算、論理演算、シフト演算およびビット演算に用いられる。

【0032】フォーマット113(Short_B1)は、演算内容を指定するフィールド120およびレジスタ番号を指定するフィールド126で構成される。このフォーマット113は、レジスタ指定によるジャンプ命令および分岐命令に用いられる。フォーマット114(Short_B2)は、演算内容を指定するフィールド120および18ビット長のディスプレイメントの

フィールド127で構成される。このフォーマット114は、ジャンプ命令および分岐命令に用いられる。

【0033】フォーマット115 (Short_B3) は、演算内容を指定するフィールド120、レジスタ番号を指定するフィールド121、レジスタ番号または12ビット長の即値を指定するフィールド128、フィールド128がレジスタ番号を示すのか即値を示すのかを指定するフィールド129、およびゼロ判定にもとづいてフィールド121にもとづく条件ジャンプまたは条件分岐を行うか否かを指定するフィールド130で構成される。このフォーマット115は、条件ジャンプ命令および条件分岐命令に使用される。

【0034】フォーマット116 (Short_D1) は、演算内容を指定するフィールド120、レジスタ番号を指定するフィールド121、レジスタ番号または12ビット長の即値を指定するフィールド128、フィールド128がレジスタ番号を示すのか即値を示すのかを指定するフィールド129で構成される。このフォーマット116は、条件ジャンプ命令、条件分岐命令およびリピート命令に使用される。フォーマット117 (Short_D2) は、演算内容を指定するフィールド120、レジスタ番号または12ビット長の即値を指定するフィールド128、フィールド128がレジスタ番号を示すのか即値を示すのかを指定するフィールド129、遅延命令 (ディレイド命令) に関するフィールド131で構成される。このフォーマット117は、ディレイドジャンプ命令、ディレイド分岐命令およびリピート命令に使用される。

【0035】フォーマット118 (Long) は、演算内容を指定するフィールド120、レジスタ番号を指定する2つのフィールド121、122、32ビット長の即値を指定するフィールド132で構成される。このフォーマット118は、複雑な算術演算、大きな即値を用いる算術演算、大きなディスプレイスメント付きレジスタ間接アドレッシングのメモリアクセス演算、大きな変位の分岐演算および絶対番地へのジャンプ命令などに使用される。

【0036】図4はマイクロプロセッサのレジスタ構成を示す説明図である。このマイクロプロセッサは、図4(a)に示すような64本の32ビット長の汎用レジスタ5、図4(b)に示すような12本の制御レジスタ150、および図4(c)に示すような2本のアキュムレータ18を持つ。R0の汎用レジスタ140の内容は常に0であり、そこへの書き込みは無視される。R62の汎用レジスタは、サブルーチンからの戻り先アドレスが設定されるリンクレジスタである。R63の汎用レジスタは、スタックポインタであり、PSW10のSMフィールドの値に応じてユーザスタックポインタ (SPU) または割り込みスタックポインタ (SPI) として動作*

A-1. ロード/ストア命令

*する。制御レジスタ150には、プログラムカウンタ151、PSW10、および各種の専用レジスタが含まれる。図3に示すフォーマット112による演算では、64本の汎用レジスタ5のそれぞれを上位16ビットと下位16ビットとに分けてアクセスできる。また、2本のアキュムレータ18を、上位32ビットと下位32ビットとに分けて別々にアクセスできる。

【0037】図5はPSW10の詳細内容を示す説明図である。PSW10の上位16ビットには、スタックポインタを切り替えるためのSMフィールド171、セルフデバッグトラップ (SDBT) の検出を示すEAフィールド172、SDBTの許可を指定するDBフィールド173、割り込み許可を指定するIEフィールド174、リピート動作の許可を指定するRPフィールド175、モジュロアドレッシングの許可を指定するMDフィールド176がある。下位16ビットはフラグフィールド180である。フラグフィールド180には8個のフラグがあり、その中のF0フラグ181およびF1フラグ182は演算の有効/無効を指定する。各フラグの値は比較演算や算術演算の結果に応じて変化する。また、フラグ初期化演算で初期化したり、フラグ値書き込み演算で任意の値をフラグフィールド180に書き込むことによって変化する。フラグフィールド180の内容は、フラグ値読み出し演算によって読み出される。

【0038】各フラグは、以下のような意味を有する。

SM=0	: スタックモード0→SPIを使用
SM=1	: スタックモード1→SPUを使用
EA=0	: SDBTを未検出
EA=1	: SDBTを検出
DB=0	: SDBTを非許可
DB=1	: SDBTを許可
IE=0	: 割り込み非許可
IE=1	: 割り込み許可
RP=0	: リピートブロック無効
RP=1	: リピートブロック有効
MD=0	: モジュロアドレッシング無効
MD=1	: モジュロアドレッシング有効
F0	: 汎用フラグ (実行コントロールフラグ)
F1	: 汎用フラグ (実行コントロールフラグ)
F2	: 汎用フラグ
F3	: 汎用フラグ
F4 (S)	: 飽和演算フラグ
F5 (V)	: オーバーフローフラグ
F6 (VA)	: 累積オーバーフローフラグ
F7 (C)	: キャリー/ボローフラグ

【0039】以下、このマイクロプロセッサの命令一覧を示す。

A. マイクロプロセッサ機能に関する命令

11	
LDB	: Load one byte to a register with sign extension [1バイトロード (符号拡張あり)]
LDBU	: Load one byte to a register with zero extension [1バイトロード (ゼロ拡張あり)]
LDH	: Load one half-word to a register with sign extension [1ハーフワードロード (符号拡張あり)]
LDHH	: Load one half-word to a register high [1ハーフワードロード (レジスタ上位へ)]
LDHU	: Load one half-word to a register with zero extension [1ハーフワードロード (ゼロ拡張あり)]
LDW	: Load one word to a register [1ワードロード]
LD2W	: Load two words to registers [2ワードロード]
LD4BH	: Load four bytes to four half-words in two registers with sign extension [4バイトロード (2レジスタへ, 符号拡張あり)]
LD4BHU	: Load four bytes to four half-words in two registers with zero extension [4バイトロード (2レジスタへ, ゼロ拡張あり)]
LD2H	: Load two half-words to two words in two registers with sign extension [2ハーフワードロード (2レジスタへ, 符号拡張あり)]
STB	: Store one byte from a register [1バイトストア]
STH	: Store one half-word from a register [1ハーフワードストア]
STHH	: Store one half-word from a register high [1ハーフワードストア (レジスタ上位から)]
STW	: Store one word from a register [1ワードストア]
ST2W	: Store two words from registers [2ワードストア]
ST4HB	: Store four bytes from four half-words from two registers [4バイトストア (2レジスタの4ハーフワードから)]
ST2H	: Store two half-words from two registers [2ハーフワードストア (2レジスタから)]
MODDEC	: Decrement a register value by a 5-bits immediate value [即値5ビットのデクリメント]
MODINC	: Increment a register value by a 5-bits immediate value [即値5ビットのインクリメント]

【0040】

A-2. 転送命令

MVFSYS	: Move a control register to a general purpose register [制御レジスタから汎用レジスタへ]
MVTSYS	: Move a general purpose register to a control register [汎用レジスタから制御レジスタへ]
MVFACC	: Move a word from an accumulator [アキュムレータからの1ワード転送]

13

14

MVTACC : Move two general purpose registers to an accumulator
[2汎用レジスタ内容のアクキュムレータへの転送]

【0041】

A-3. 比較命令

CMPcc : Compare [比較]
cc = EQ (等しい), NE (不等), GT (より大),
GE (以上), LT (未満), LE (以下),
PS (ともに正), NG (ともに負)
CMPUcc : Compare unsigned [比較 (符号なし)]
cc = GT, GE, LT, LE

【0042】A-4. 最大値/最小値命令

*【0043】

reserved [予約済]

*

A-5. 算術演算命令

ABS : Absolute [絶対値をとる]
ADD : Add [加算]
ADDC : Add with carry [加算 (キャリー付き)]
ADDHppp : Add half-word [ハーフワード加算]
ppp = LLL (レジスタ下位, レジスタ下位, レジスタ下位), LLH (レジスタ下位, レジスタ下位, レジスタ上位), LHL, LHH, HLL, HLH, HHL, HHH
ADDS : Add register Rb with the sign of the third operand
[レジスタRbに符号を付ける]
ADDS2H : Add sign to two half-words
[2ハーフワードに符号を付ける]
ADD2H : Add two pairs of half-words
[2ハーフワード同士の加算]
AVG : Average with rounding towards positive infinity
[平均をとる]
AVG2H : Average two pairs of half-words rounding
towards positive infinity
[2ハーフワードそれぞれの平均をとる]
JOINpp : Join two half-words [2ハーフワードの結合]
pp = LL, LH, HL, HH
SUB : Subtract [減算]
SUBB : Subtract with borrow [ボロ付き減算]
SUBHppp : Subtract half-word [ハーフワードの減算]
ppp = LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH
SUB2H : Subtract two pairs of half-words
[2つのハーフワードの減算]

【0044】

A-6. 論理演算命令

AND : logical AND [論理積]
OR : logical OR [論理和]
NOT : logical NOT [反転]
XOR : logical exclusive OR [排他的論理和]
ANDFG : logical AND flags [フラグの論理積]
ORFG : logical OR flags [フラグの論理和]
NOTFG : logical NOT a flag [フラグの反転]

15

16

XORFG : logical exclusive OR flags [フラグの排他的論理和]

【0045】

A-7. シフト演算命令

SRA : Shift right arithmetic [算術右シフト]

SRA2H : Shift right arithmetic two half-words
[2つのハーフワードの算術右シフト]SRC : Shift right concatenated registers
[レジスタ連鎖右シフト]

SRL : Shift right logical [論理右シフト]

SRL2H : Shift right logical two half-words
[2つのハーフワードの論理右シフト]

ROT : Rotate right [右回転]

ROT2H : Rotate right two half-words
[2つのハーフワードの右回転]

【0046】 A-8. ビット操作命令

* BSET : Set a bit [ビットセット]

BCLR : Clear a bit [ビットクリア]

BTST : Test a bit [ビットテスト]

BNOT : Invert a bit [ビット反転]

* 【0047】

A-9. 分岐命令

BRA : Branch [分岐]

BRATZR : Branch if zero [ゼロなら分岐]

BRATNZ : Branch if not zero [ゼロでないなら分岐]

BSR : Branch to subroutine [サブルーチンへ分岐]

BSRTZR : Branch to subroutine if zero
[ゼロならサブルーチンへ分岐]BSRTNZ : Branch to subroutine if not zero
[ゼロでないならサブルーチンへ分岐]

JMP : Jump [無条件ジャンプ]

JMPTZR : Jump if zero [ゼロならジャンプ]

JMPTNZ : Jump if not zero [ゼロでないならジャンプ]

JSR : Jump to subroutine [サブルーチンへジャンプ]

JSRTZR : Jump to subroutine if zero
[ゼロならサブルーチンへジャンプ]JSRTNZ : Jump to subroutine if not zero
[ゼロでないならサブルーチンへジャンプ]

NOP : No Operation [無操作]

[ディレイド分岐、ジャンプ命令]

DBRA : Delayed branch [ディレイド分岐]

DBRAI : Delayed branch immediate [ディレイド分岐(即値)]

DBSR : Delayed branch to subroutine

e

[ディレイドサブルーチン分岐]

DBSRI : Delayed branch immediate to
subroutine

[ディレイドサブルーチン分岐(即値)]

DJMP : Delayed jump [ディレイドジャンプ]

DJMPI : Delayed jump immediate [ディレイドジャンプ(即値)]

DJSR : Delayed jump to subroutine
[ディレイドサブルーチンジャンプ]DJSRI : Delayed jump immediate to subroutine
[ディレイドサブルーチンジャンプ(即値)]

【0048】

A-10. OS関連命令

TRAP : Trap [トラップ]
 REIT : Return from exception, interrupts and traps
 [例外、割り込み、トラップからのリターン]

【0049】

B. DSP機能に関する命令

B-1. 算術操作命令

MUL : Multiply [乗算]
 MULX : Multiply with extended precision [倍精度乗算]
 MULXS : Multiply and shift to the right by one
 with extended precision
 [倍精度乗算および1ビット右シフト]
 MULX2H : Multiply two pairs of half-words
 with extended precision
 [2ハーフワードずつの倍精度乗算]
 MULHXpp : Multiply two half-words with extended precision
 pp=LL, LH, HL, HH
 [2ハーフワードの倍精度乗算]
 MUL2H : Multiply two pairs of half-
 words
 [2ハーフワードずつの乗算]
 MACa : Multiply and add [積和演算]
 a (アキュムレータ指定) = 0, 1
 MACSa : Multiply, shift to the right by one and add
 a = 0, 1
 [1ビット右シフト付き積和演算]
 MSUBa : Multiply and subtract [積和(減算)演算]
 a = 0, 1
 MSUBSa : Multiply, shift to the right by one and subtract
 a = 0, 1
 [1ビット右シフト付き積和(減算)演算]

【0050】

B-2. リピート命令

REPEAT : Repeat a block of instructions
 [命令ブロックの繰り返し]
 REPEAT I : Repeat a block of instructions immediate
 [命令ブロックの繰り返し(即値指定)]

【0051】図6はマイクロプロセッサの並列2命令実行時のパイプライン動作を示す説明図である。この動作は、命令のフォーマットフィールドの値FM=00のときに実行される。パイプライン190、195は、命令フェッチステージ191、デコード/アドレス演算ステージ192、実行/メモリアクセスステージ193およびライトバックステージ194で構成される。並列2命令実行時には、メモリユニット3での実行と整数演算ユニット4での実行とが並列に行われる。図7はマイクロプロセッサのシーケンシャル命令実行時のパイプライン動作を示す説明図である。この動作は、命令のフォーマットフィールドの値FM=01、10、11のときに実

40 *行される。パイプライン200は、命令フェッチステージ、デコード/アドレス演算ステージ、実行/メモリアクセスステージ、およびライトバックステージで構成されるが、この場合には、メモリユニット3での実行と整数演算ユニット4での実行とのうちのいずれかが、一時に実行される。なお、図6および図7に示されたパイプライン動作では、2番目のステージであるデコード/アドレス演算ステージ192において分岐先アドレスが得られる。

【0052】図8はこの発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。図において、366は分岐命令のデコード値を保持するための

レジスタ13Aと分岐命令が実行されるべき時期に対応したPC値を保持するレジスタ13Bとを含む遅延分岐命令レジスタ(遅延分岐命令保持手段)である。分岐命令が実行されるべき時期に対応したPC値とは、マイクロプロセッサのPCの値がその値になるとディレイド分岐命令を実行することになる値である。なお、既に定義したように、単に「分岐命令」と表現した場合には、厳密な意味での分岐命令とジャンプ命令との双方の概念を含む。

【0053】図9はディレイド分岐命令の基本的なフォーマット320を示す説明図である。基本的には、ディレイド分岐命令のフォーマット320は、オペコード321、遅延量を指定するフィールド322および分岐先アドレスを指定するためのオフセットまたはアドレスが指定されるフィールド323を持つ。ディレイド分岐命令は、例えば、図3に示すフォーマット116(Short_D1)、フォーマット117(Short_D2)またはフォーマット118(Long)で実現される。フォーマット116(Short_D1)は、遅延量としてレジスタ設定値が用いられる場合に使用される。フォーマット117(Short_D2)は、遅延量として即値が用いられる場合に使用される。フォーマット118(Long)は、分岐先アドレスを32ビット即値で指定する場合に使用される。各フォーマットにおいて、オペコードはフィールド120で指定される。また、フィールド129は、フィールド128がレジスタ番号を示すのか即値を示すのかを指定するために使用される。フィールド121は、DBRA、DBSR、DJMP、DJSRの各命令において遅延量がレジスタで指定されるときにレジスタ指定領域として使用され、フィールド131は、遅延量を指定する即値の領域として使用される。

【0054】図10は幾つかのディレイド分岐命令の例を示す説明図である。命令324は、遅延量を即値で指定するとともに分岐のオフセットを即値で指定する命令である。命令325は、遅延量をレジスタで指定するとともに分岐のオフセットを即値で指定する命令である。命令326は、遅延量を即値で指定するとともに分岐のオフセットをレジスタで指定する命令である。命令327は、遅延量をレジスタで指定するとともに分岐のオフセットをレジスタで指定する命令である。

【0055】図11はディレイド分岐命令330、ディレイドジャンプ命令332およびディレイドサブルーチンコール命令331、333が、同一フォーマットでどのように実現されるのかを示すための説明図である。

【0056】次に動作について説明する。命令がディレイド分岐命令であったことを命令デコードユニット2のデコーダ8が認識すると、命令デコードユニット2は、そのことを示す制御信号11をメモリユニット3に出力する。メモリユニット3において、PC制御部13は、

制御信号11によって入力した命令デコード結果をレジスタ13Aに格納する。従って、レジスタ13Aには、ディレイド分岐命令による分岐先を示す情報等が格納される。また、PC制御部13は、分岐命令が実行されるべき時期に応じたPC値をレジスタ13Bに格納する。

【0057】PC制御部13は、マイクロプロセッサにおける実際のPC値がレジスタ13Bに格納されている値と一致したことを検知したら、レジスタ13Aに格納されている分岐先情報にもとづいて分岐命令を実行する。すなわち、レジスタ13Aに格納されている分岐先情報が示す値をPCに設定する。この結果、レジスタ13Bに格納されている値に応じた数の命令がフェッチされたら、次のサイクルで、分岐先にある命令がフェッチされる。

【0058】図12はディレイド分岐命令を含むプログラムの一例を示す説明図である。図12における2行目のDBRA命令は、遅延量「3」を指定するとともに分岐先としてTGT1を指定する命令である。命令デコードユニット2において、DBRA命令がデコードされると、PC制御部13は、分岐先を示すTGT1に応じた情報をレジスタ13Aに格納する。また、遅延量「3」に応じたPC値、すなわち、そのときのPCの値に3×8を加算した値をレジスタ13Bに格納する。DBRA命令の下に記述されている3命令は無条件に処理されるが、3番目の遅延スロットにある命令をフェッチするとき、マイクロプロセッサのPCの値はレジスタ13Bに格納されているPC値に一致する。そこで、PC制御部13は、DBRA命令による分岐処理を実行する。すなわち、レジスタ13Bに格納されているPC値をマイクロプロセッサにおけるPCに設定する処理を行う。

【0059】図13はディレイド分岐命令を含むプログラムの他の例を示す説明図である。ここでは、サブルーチンコール命令であるDJSR命令が用いられている。また、遅延量は、汎用レジスタ5におけるr4レジスタに設定されている値が用いられる。この場合も、PC制御部13は、分岐先を示すTGT1に応じた情報をレジスタ13Aに格納する。また、PC制御部13は、r4レジスタに設定されている値に応じたPC値をレジスタ13Bに格納する。r4レジスタに設定されている値が「4」であったとすると、DJSR命令の下に記述されている4命令は無条件に処理されるが、4番目の遅延スロットの命令をフェッチするときに、マイクロプロセッサのPCの値はレジスタ13Bに格納されているPC値に一致する。そこで、PC制御部13は、DJSR命令によるサブルーチンジャンプを実行する。具体的には、PCに、レジスタ13Bに格納されているPC値を設定する。

【0060】以上のように、この実施の形態1によれば、マイクロプロセッサは、ディレイド分岐命令で指定された遅延量に応じたPC値を保持するように構成され

21

ているので、ディレイド分岐命令のデコード時点から実行時点までの間に割り込み等のPC値を変化させる事象が生じたとしても、確実に分岐命令が実行される。例えば、図12に示された例において、遅延スロットの命令の実行時に割り込みが生ずるとマイクロプロセッサのPCの値は変化するとともに、割り込み処理において幾つかのサイクルが消費される。従来のマイクロプロセッサにおける処理のようにディレイド分岐命令の実行時期をカウンタ値として保持していると、割り込み処理において費やされたサイクル数分だけ実行時期がずれてしまう。しかし、この実施の形態によれば、そのようなことは無い。また、レジスタ13A、13Bをアクセスする命令を備えれば、遅延スロットの命令実行時に起きた割り込みにもとないコンテキストスイッチが必要になった場合、レジスタ13Aに保持された分岐先情報とレジスタ13Bに保持された分岐発生PC値とをデータRAM7に退避し、異なるコンテキストを実行後、再び割り込まれたコンテキストに復帰したときにこれらの値を復帰して遅延分岐を実行することも可能になる。

【0061】実施の形態2. 図14はこの発明の実施の形態2によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。図において、361は算術論理演算を実行するALU（演算ユニット）、363は乗算を実行する乗算器（演算ユニット）、365はPC値を計算するPC制御部、367はアドレス計算を行うメモリ制御部（演算ユニット）、369はシフト演算を実行するシフト（演算ユニット）、371は1サイクルで2命令（2演算指令を含む命令）を転送できるバス、372は命令をデコードして命令実行部に制御信号11、12を与える命令デコーダ、373は汎用レジスタである。

【0062】ALU361において、362はディレイド算術論理演算命令のデコード結果を格納するレジスタ362Aと遅延量に応じた値を格納するレジスタ362Bとを有する遅延命令レジスタ（遅延命令保持手段）である。乗算器363において、364はディレイド乗算命令のデコード結果を格納するレジスタ364Aと遅延量に応じた値を格納するレジスタ364Bとを有する遅延命令レジスタ（遅延命令保持手段）である。PC制御部365において、366はディレイド分岐命令のデコード結果を格納するレジスタ366Aと遅延量に応じた値を格納するレジスタ366Bとを有する遅延命令レジスタ（遅延分岐命令保持手段）である。メモリ制御部367において、368はディレイドメモリアクセス命令のデコード結果を格納するレジスタ368Aと遅延量に応じた値を格納するレジスタ368Bとを有する遅延命令レジスタ（遅延命令保持手段）である。シフト369において、370はディレイドシフト命令のデコード結果を格納するレジスタ370Aと遅延量に応じた値を格納するレジスタ370Bとを有する遅延命令レジスタ

22

（遅延命令保持手段）である。

【0063】次に動作について説明する。図15は各行の2演算を同時に実行する命令配置の一例を示す説明図である。この配置は、図22に示されたプログラムをスケジューリングした結果である。図21に示された一般的な2演算を同時に扱うマイクロプロセッサでは、図22に示されたプログラムを実行するために、図23に示されたように5サイクルを要していた。SRA命令とSUB命令とはr3レジスタ依存の関係にあって同時に実行できないからである。ところが、この実施の形態によるマイクロプロセッサはディレイド分岐命令を扱えるので、図15に示されたように、図23における2行目のNOPを遅延量「2」のDJMP命令で置き換えることができる。

【0064】図15に示されたような命令配置による各命令を順次入力すると、第1サイクルでは、マイクロプロセッサにおいて、ALU361とシフト369とが、1行目のADD演算とSRA演算とをそれぞれ実行する。第2サイクルでは、シフト369とPC制御部365とがSRA演算とDJMP演算とを扱う。シフト369はSRA演算を直ちに実行するが、PC制御部365は、実施の形態1の場合と同様に、DJMP演算のデコード結果である分岐先TGTに関する情報をレジスタ366Aに格納し、遅延量「2」に応じたPC値をレジスタ366Bに格納する。第3サイクルでは、ALU361と乗算器363とが、3行目のSUB演算とMUL演算とをそれぞれ実行する。第4サイクルでは、ALU361と乗算器363とが、4行目のADD演算とMUL演算とをそれぞれ実行する。4行目の命令がフェッチされるとき、マイクロプロセッサのPC（図14において図示せず）の値はレジスタ366Bに格納されているPC値に一致している。そこで、PC制御部365は、DJMP演算で指定されたジャンプ命令を実行する。具体的には、PCに、レジスタ366Bに格納されているPC値を設定する。

【0065】このように、遅延量を任意に指定できる分岐命令処理する機構を備えていれば、命令のスケジューリングをより有効に実行することができる。すなわち、プログラマまたはコンパイラは、プログラムサイズをより小さくするような命令のスケジューリングを行える。その結果、ある処理を実行するのに要する時間は短縮される。しかも、PC制御部365は、分岐命令実行時期をPC値として保持するので、ディレイド分岐命令は矛盾なく実行される。

【0066】この実施の形態によるマイクロプロセッサは、ディレイド分岐命令以外のディレイド演算命令を扱うこともできる。図16は、プログラムの一例を示す説明図である。図21に示された一般的な2演算を同時に扱うマイクロプロセッサでは、図16における1行目のADD演算と2行目のADD演算を同時実行することは

できない。1つのALU341しか持っていないからである。従って、図16における1行目の命令と2行目の命令とを同時実行するように命令のスケジューリングを行うことはできなかった。

【0067】ところが、この実施の形態によるマイクロプロセッサは、図16に示すプログラムの各命令を図17に示すようにスケジューリングした各命令を実行できる。図17に示されたような命令配置による各命令を順次入力すると、このマイクロプロセッサにおいて、第1サイクルでは、ALU361は、ADD演算を実行するとともにDADD（ディレイドADD）演算も扱う。すなわち、DADDのデコード結果をレジスタ362Aに格納するとともに、遅延値「1」に応じた値をレジスタ362Bに格納する。第2サイクルでは、シフト369と乗算器363とは、SRA演算とMUL演算とをそれぞれ実行する。2行目の命令がフェッチされるときに、PCの値はレジスタ362B内の遅延値「1」に応じた値に一致する。そこで、ALU361は、レジスタ362AからDADDで指定された演算指令内容を取り出し、その演算を実行する。

【0068】なお、レジスタ362Bに格納される遅延値に応じた値は、動作クロックに応じて値が変化するカウンタ値であってもよいが、PC値であってもよい。遅延値に応じた値としてPC値を保持した場合には、デコード時と実行時との間で割り込み等が生じてもそれに対する対処が容易になる。

【0069】ここでは、ディレイド演算命令を扱う演算ユニットとしてALU361を例にとったが、乗算器363、メモリ制御部367およびシフト369も、ALU361の処理と同様の処理によってディレイド演算命令を扱うことができる。また、遅延値「1」の場合を例に説明したが、もちろん、任意の値を指定できる。任意の値を指定可能なので、命令のスケジューリングの自由度をより向上させることができる。なお、固定的な値を指定するディレイド演算命令しか処理しないマイクロプロセッサであっても、任意の遅延値を指定できるディレイド演算命令を処理するマイクロプロセッサに比べると効果は低減するものの、命令のスケジューリングの自由度を向上させる効果が期待できる。

【0070】以上のように、ディレイド演算命令を扱う機構を有する場合には、プログラムの実行に要するサイクル数をさらに少なくでき、その結果、プログラム実行時間を短縮できる。例えば、図16に示されたプログラムがあった場合に、図21に示された一般的な2演算を同時に扱うマイクロプロセッサでは、2つのADD命令を同時実行できないので、命令のスケジューリングを行ってもプログラム実行に3サイクルを要する。このマイクロプロセッサは、図17に示された例からわかるように、2サイクルでプログラムを実行できる。また、ルーブ内でディレイド命令を記述できるような場合に、この

マイクロプロセッサによれば、さらに処理を高速化できる。

【0071】ディレイド命令は、2演算同時実行できるマイクロプロセッサにおいて、特に有効である。図15および図17に示された例からわかるように、演算を並列実行する場合のNOP挿入箇所を削減できるからである。なお、ここでは、2演算同時実行のマイクロプロセッサについて説明したが、リソースであるALU361等の演算ユニットを複数個備え、同時実行演算数をさらに多くしたマイクロプロセッサに、この実施の形態によるディレイド演算命令処理機構を設けてもよい。そのようなマイクロプロセッサにおいても、実行サイクル数の削減および処理の高速化が期待できる。

【0072】図18はこの実施の形態2によるマイクロプロセッサにおけるPSWの一例を示す説明図である。このPSW380、390は図5に示されたものと同様のものであるが、この場合には、RPフィールド381がPC制御部365の遅延命令レジスタ366の動作を有効にするかどうか決めるビットとして用いられる。また、例えば、E4フィールド382、E3フィールド383、E2フィールド384およびE1フィールド385は、それぞれ、ALU361の遅延命令レジスタ362、乗算器363の遅延命令レジスタ364、メモリ制御部367の遅延命令レジスタ368およびシフト369の遅延命令レジスタ370の動作を有効にするかどうか決めるビットとして用いられる。各ビットが無効状態にセットされた場合には、対応する演算ユニットにおけるディレイド命令処理は禁止される。このようなPSW380、390を用意することによって、ディレイド命令処理の柔軟性を増すことができる。

【0073】実施の形態3. 図19はこの発明の実施の形態3によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。図において、461は算術論理演算を実行するALU（演算ユニット）、463は乗算を実行する乗算器（演算ユニット）、465はPC値を計算するPC制御部、467はアドレス計算を行うメモリ制御部（演算ユニット）、469はシフト演算を実行するシフト（演算ユニット）、371は1サイクルで2命令を転送できるバス、372は命令をデコードして命令実行部に制御信号11、12を与える命令デコード、373は汎用レジスタである。

【0074】このマイクロプロセッサにおいて、ALU461は、2つの遅延命令レジスタ362を有する。乗算器463は、2つの遅延命令レジスタ364を有する。PC制御部465は、2つの遅延命令レジスタ366を有する。メモリ制御部467は、2つの遅延命令レジスタ368を有する。そして、シフト469は、2つの遅延命令レジスタ370を有する。

【0075】次に動作について説明する。各演算ユニッ

10

20

30

40

50

トおよびPC制御部465において、命令のデコード結果と遅延量とからなる1セットはキュー管理される。例えば、ALU461において、一方の遅延命令レジスタ362のレジスタ362Aにディレイド算術演算命令のデコード結果が格納されレジスタ362Bにそれに対応する遅延量に応じた値が設定されている場合に、さらに、命令デコーダ372から新たなディレイド算術演算命令のデコード結果が送られてきたとする。すると、新たなデコード結果および遅延量に応じた値は、他方の遅延命令レジスタ362に格納される。そして、一方のレジスタ362Bに格納されている遅延量に応じた値とマイクロプロセッサのPCの値とが一致したときに、一方のレジスタ362Aに格納されているデコード結果に応じた演算が実行される。その後、他方のレジスタ362Bに格納されている遅延量に応じた値とマイクロプロセッサのPCの値とが一致したときに、他方のレジスタ362Aに格納されているデコード結果に応じた演算が実行される。

【0076】そのような構成によれば、あるディレイド命令の実行時期に到達していない時点でも、別の同種類のディレイド命令を扱うことができる。例えば、あるディレイドADD命令の実行時期に到達していない時点でも、ALU461は、新たなディレイドADD命令を受け入れることができる。従って、命令のスケジューリングの自由度をさらに上げることができ、プログラムサイズの減少すなわち命令メモリの容量削減を図ることができる。また、より高速にプログラムを実行することができる。なお、ここでは各演算ユニットにおいて2つの遅延命令レジスタが設けられている例を示したが、設置数をさらに多くしてもよい。

【0077】

【発明の効果】以上のように、請求項1記載の発明によれば、遅延命令を有するマイクロプロセッサを、ディレイド分岐命令で指定された遅延量に応じたプログラムカウンタ値を保持するように構成したので、ディレイド分岐命令のデコード時点から実行時点までの間に割り込み等のPCの値を変化させる事象が生じたとしても確実に分岐命令が実行される効果がある。

【0078】請求項2記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットがディレイド演算命令の固定のまたは可変の遅延量に応じた値を保持するように構成したので、命令スケジューリングの自由度を向上させることができる効果がある。

【0079】請求項3記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットがディレイド演算命令で指定された任意の遅延量に応じた値を保持するように構成したので、プログラムの実行に要するサイクル数をより少なくでき、その結果、命令メモリの容量を削減できるとともにプログラム実行時間を短縮できる効果がある。

【0080】請求項4記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットがディレイド演算命令で指定された遅延量に応じたPC値を保持するように構成したので、プログラムの実行に要するサイクル数をより少なくできるとともに、ディレイド演算命令のデコード時点から実行時点までの間に割り込み等のPCの値を変化させる事象が生じたとしても確実に演算が実行される効果がある。

【0081】請求項5記載の発明によれば、遅延命令を有するマイクロプロセッサを、各演算ユニットに複数の遅延命令保持手段が設けられるように構成したので、命令のスケジューリングの自由度を上げることができる効果がある。

【0082】請求項6記載の発明によれば、遅延命令を有するマイクロプロセッサを、命令実行部が複数演算を同時に実行するように構成したので、命令のスケジューリングの自由度をさらに上げることができ、その結果、プログラムの実行に要するサイクル数をさらに少なくできる効果がある。

【図面の簡単な説明】

【図1】 この発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。

【図2】 マイクロプロセッサの命令フォーマットを示す説明図である。

【図3】 演算フィールドの詳細な内容を示す説明図である。

【図4】 マイクロプロセッサのレジスタ構成を示す説明図である。

【図5】 PSWの詳細内容を示す説明図である。

【図6】 マイクロプロセッサの並列2命令実行時のパイプライン動作を示す説明図である。

【図7】 マイクロプロセッサのシーケンシャル命令実行時のパイプライン動作を示す説明図である。

【図8】 この発明の実施の形態1によるマイクロプロセッサの構成を示すブロック図である。

【図9】 ディレイド分岐命令の基本的なフォーマットを示す説明図である。

【図10】 幾つかのディレイド分岐命令の例を示す説明図である。

【図11】 ディレイド分岐命令、ディレイドジャンプ命令およびディレイドサブルーチンコール命令が、同一フォーマットでどのように実現されるのかを示すための説明図である。

【図12】 ディレイド分岐命令を含むプログラムの一例を示す説明図である。

【図13】 ディレイド分岐命令を含むプログラムの他の例を示す説明図である。

【図14】 この発明の実施の形態2によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。

10

20

30

40

50

27

【図15】 2演算を同時に実行する命令配置の一例を示す説明図である。

【図16】 プログラムの一例を示す説明図である。

【図17】 図16に示された各命令にもとづく2演算を同時に実行する命令配置を示す説明図である。

【図18】 この発明の実施の形態2によるマイクロプロセッサにおけるPSWの一例を示す説明図である。

【図19】 この発明の実施の形態3によるマイクロプロセッサにおける命令デコード部分および命令実行部分を示すブロック図である。

【図20】 従来のパイプライン制御方式によるマイクロプロセッサの処理シーケンスを示すシーケンス図である。

【図21】 2演算を同時に行うマイクロプロセッサにおける命令デコードおよび命令実行部の部分の一般的な

28

構成を示すブロック図である。

【図22】 プログラムの一例を示す説明図である。

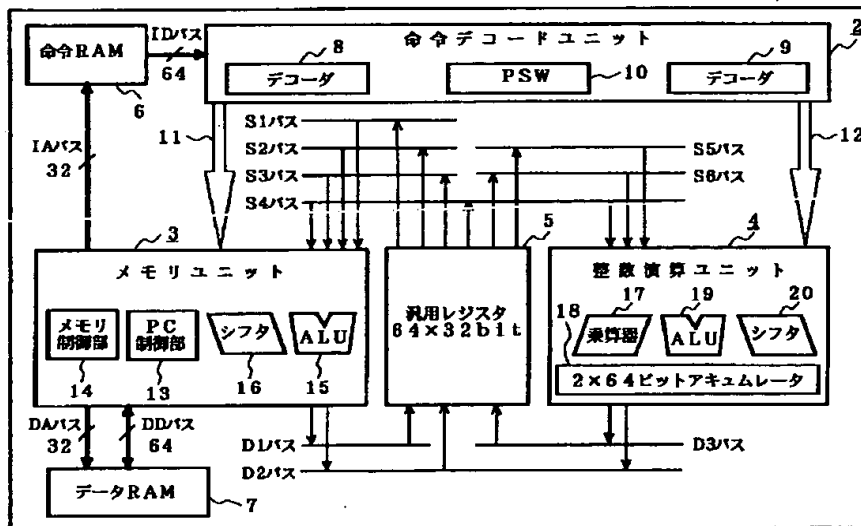
【図23】 命令のスケジューリングの一例を示す説明図である。

【符号の説明】

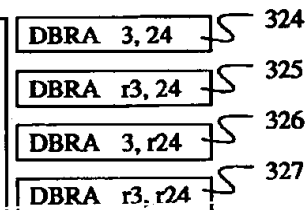
2 命令デコードユニット(命令デコーダ)、3 メモリユニット(命令実行部)、4 整数演算ユニット(命令実行部)、13、365、465 PC制御部、361、461 ALU(演算ユニット)、362、364、368、370 遅延命令レジスタ(遅延命令保持手段)、363、463 乗算器(演算ユニット)、366 遅延分岐命令レジスタ(遅延分岐命令保持手段)、367、467 メモリ制御部(演算ユニット)、369、469 シフト(演算ユニット)。

【図1】

【図10】



2:命令デコードユニット(命令デコーダ) 3:メモリユニット(命令実行部) 4:整数演算ユニット(命令実行部)

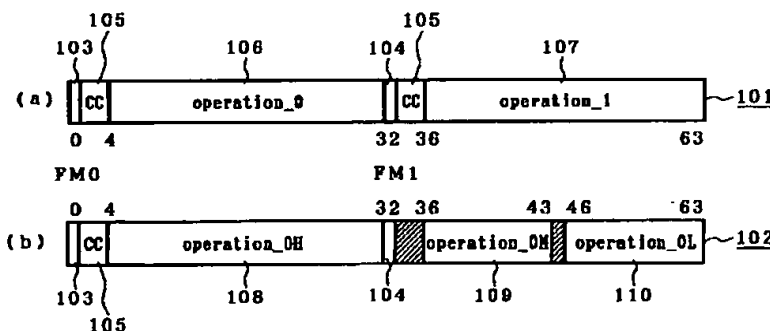


【図16】

```
ADD r3, r10, 6
ADD r4, r4, 8
SRA r8, r3, 1
MUL r5, r3, 4
```

【図2】

【図15】

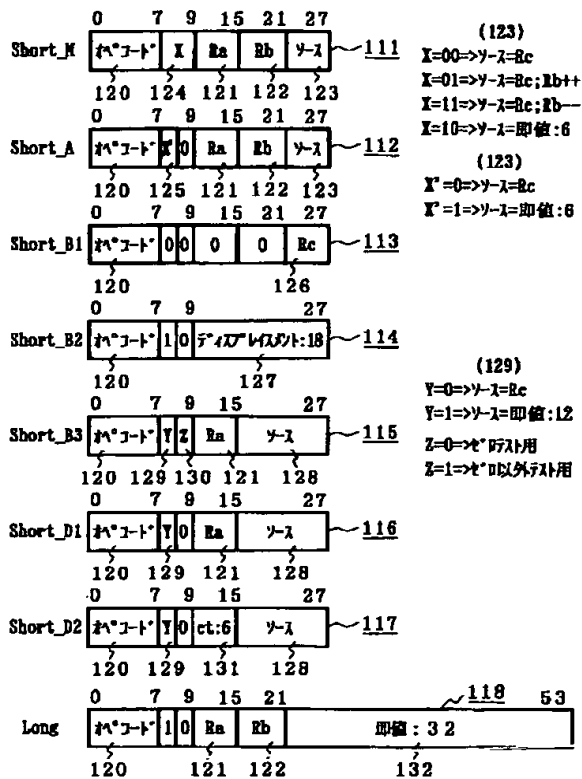


```
ADD r3, r0, 6    SRA r4, r0, 8
SRA r3, r3, 1    DJMP 2, TGT
SUB r4, r4, r3    MUL r5, r3, 4
ADD r4, r4, r10  MUL r11, r11, r5
```

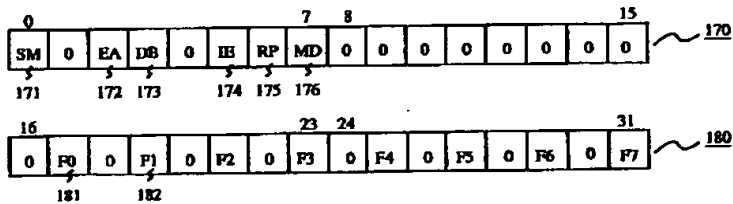
【図17】

```
ADD r3, r10, 6  DADD 1, r4, r4, 8
SRA r8, r3, 1  MUL r5, r3, 4
```

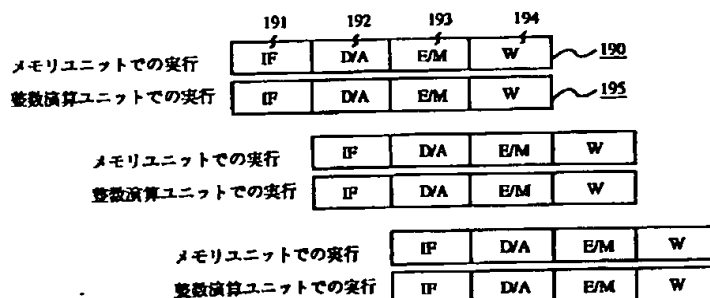
【図3】



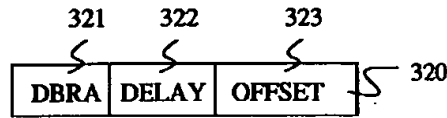
【図5】



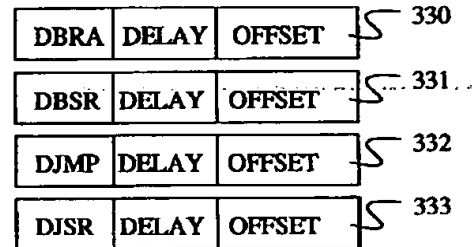
【図6】



【図9】



【☒ 1 1】



【图 2 2】

```
ADD r3, r0, 6
SRA r4, r0, 8
SRA r3, r3, 1
SUB r4, r4, r3
MUL r5, r3, 4
ADD r4, r4, r10
MUL r11, r11, r5
JMP TGT
```

【图23】

```
ADD r3, r0, 6      SRA r4, r0, 8
SRA r3, r3, 1      NOP
SUB r4, r4, r3      MUL r5, r3, 4
ADD r4, r4, r10     MUL r11, r11, r5
JMP TGT
```

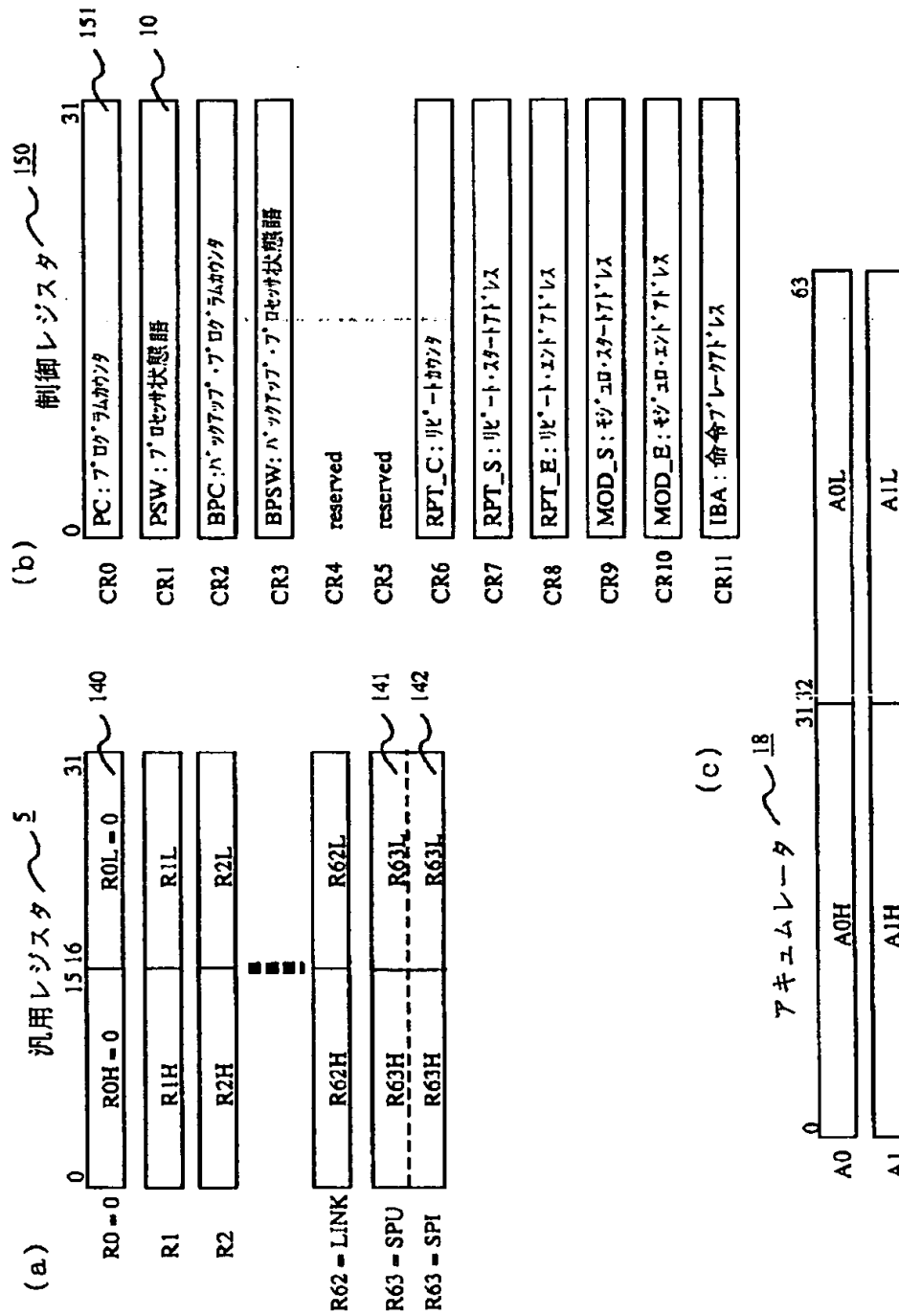
【例 12】

```

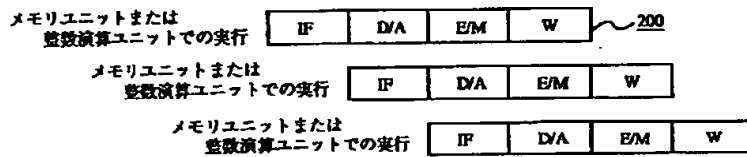
    ADD    r3, r4, r5
    DBRA   3, TGT1
    SUB    r3, r4, 10  ← delay slot 1
    SRA    r1, r2, r3  ← delay slot 2
    ADD    r3, r4, 11  ← delay slot 3
    TGT1:  ADD    r3, r4, 20
    ...

```

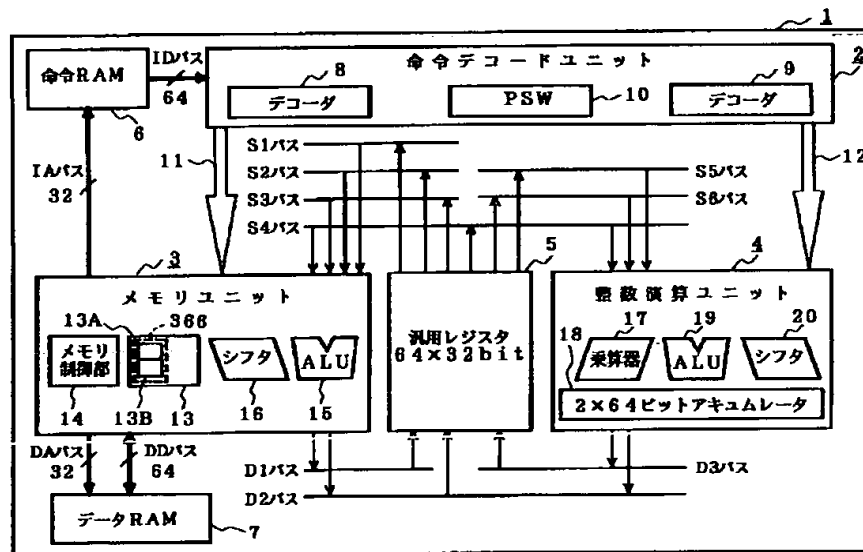

【図4】



【図7】



【図8】



366: 遅延分岐命令レジスタ (遅延分岐命令保持手段)

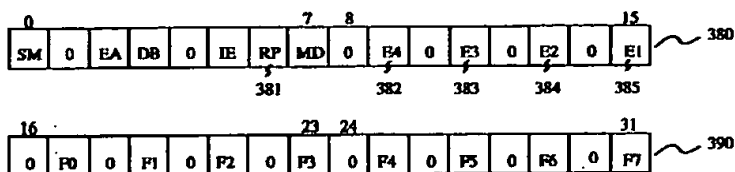
【図13】

```

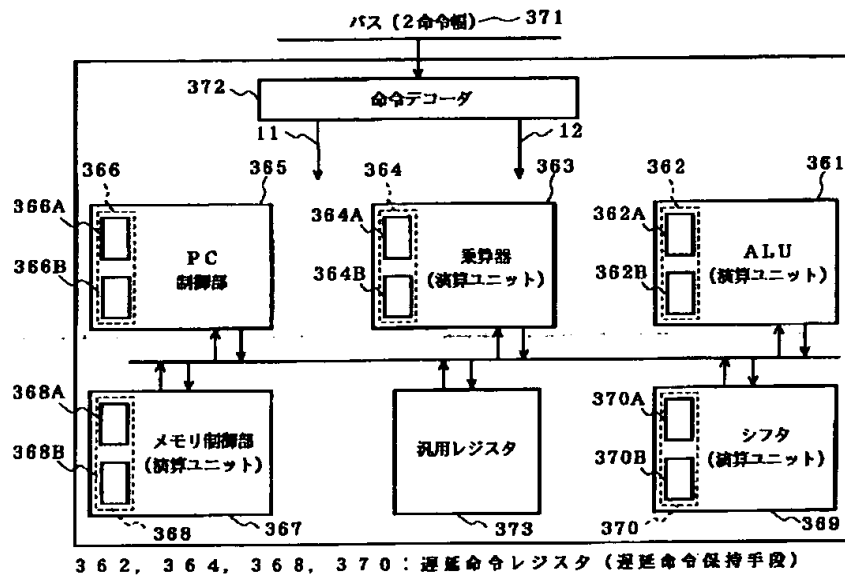
ADD    r4, r0, 4
ADD    r3, r0, 5
DJSR   r4, TGT1
SUB     r3, r4, 10 ← delay slot 1
SRA     r1, r2, r3 ← delay slot 2
ADD     r3, r4, 11 ← delay slot 3
ADD     r8, r8, 2 ← delay slot 4
...
TGT1:  ADD    r3, r4, 20
...
RSR

```

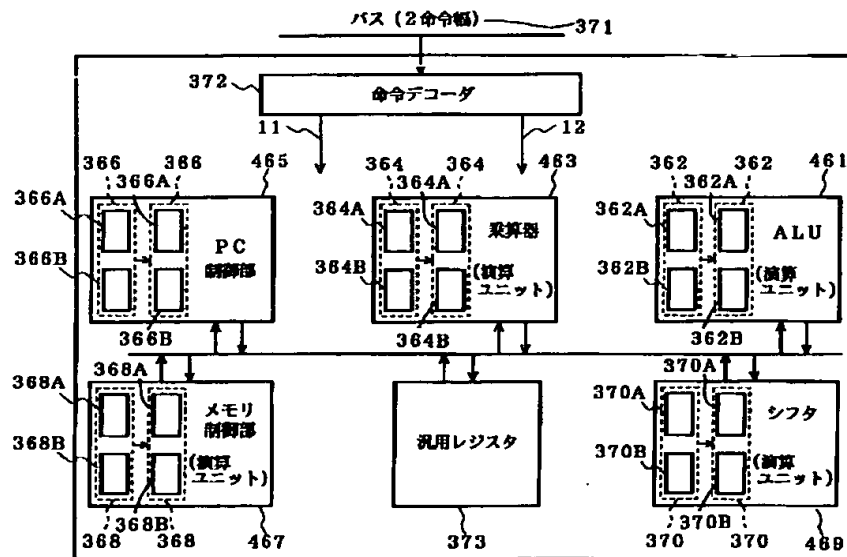
【図18】



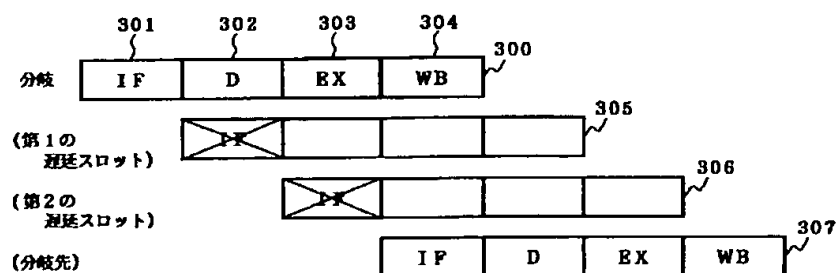
【図14】



【図19】



【図20】



【図21】

